

Figure 1

```

Itemlist items;           // each basket will have its own item list

/* @Operation
 * @Conversation Start */
void newBasket() { ... }

/* @Operation
 * @Conversation Continue */
void addItem(Item i) { ... }

/* @Operation
 * @Conversation Finish */
void checkout(Payment p) { ... }

/* @Operation
 * @Conversation Finish
 * @buffer */
void cancel() { ... }

/* @Operation */
int activeBaskets() { ... }

```

Figure 2

```
/**
 * @operation
 */
public String greeting(String firstname, String lastname)
{
    Calendar cal =
        Calendar.getInstance(TimeZone.getDefault());
    SimpleDateFormat sdf =
        new java.text.SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

    return "Hello " + firstname + " " + lastname +
        " at " + sdf.format(cal.getTime());
}
```

Figure 3A

```
POST /app/mypackage/CreditReport.jws HTTP/1.1
Host: www.company.com
Content-Type: text/xml; charset="utf-8"
Content-Length: 648
SOAPAction: "Some-URI"

<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:cgc="http://bea.com/SOAP/conversation"
  soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <m:greeting xmlns:m="http://www.company.com/app/mypackage/CreditReport.jws">
      <firstname>Jane</firstname>
      <lastname>Doe</lastname>
    </m:greeting>
  </soap:Body>
</soap:Envelope>
```

Figure 3B

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: 574

<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <m:greetingResponse
      xmlns:m="http://www.company.com/app/mypackage/CreditReport.jws">
      <result>Hello Jane Doe at 2001-08-15 16:18:04</result>
    </m:greetingResponse>
  </soap:Body>
</soap:Envelope>
```

Figure 3C

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:c="http://openuri.org/soap/conversation/">
  <s:Header>
    <c:conversationID>4232-133214-3223132:42332-3144-2332251</c:conversationID>
    <c:defaultCallbackURL>
      http://original.com/foo/Callbacks.jwi
    </c:defaultCallbackURL>
  </s:Header>
  <s:Body>
    <!-- contents omitted -->
  </s:Body>
</s:Envelope>
```

Figure 4

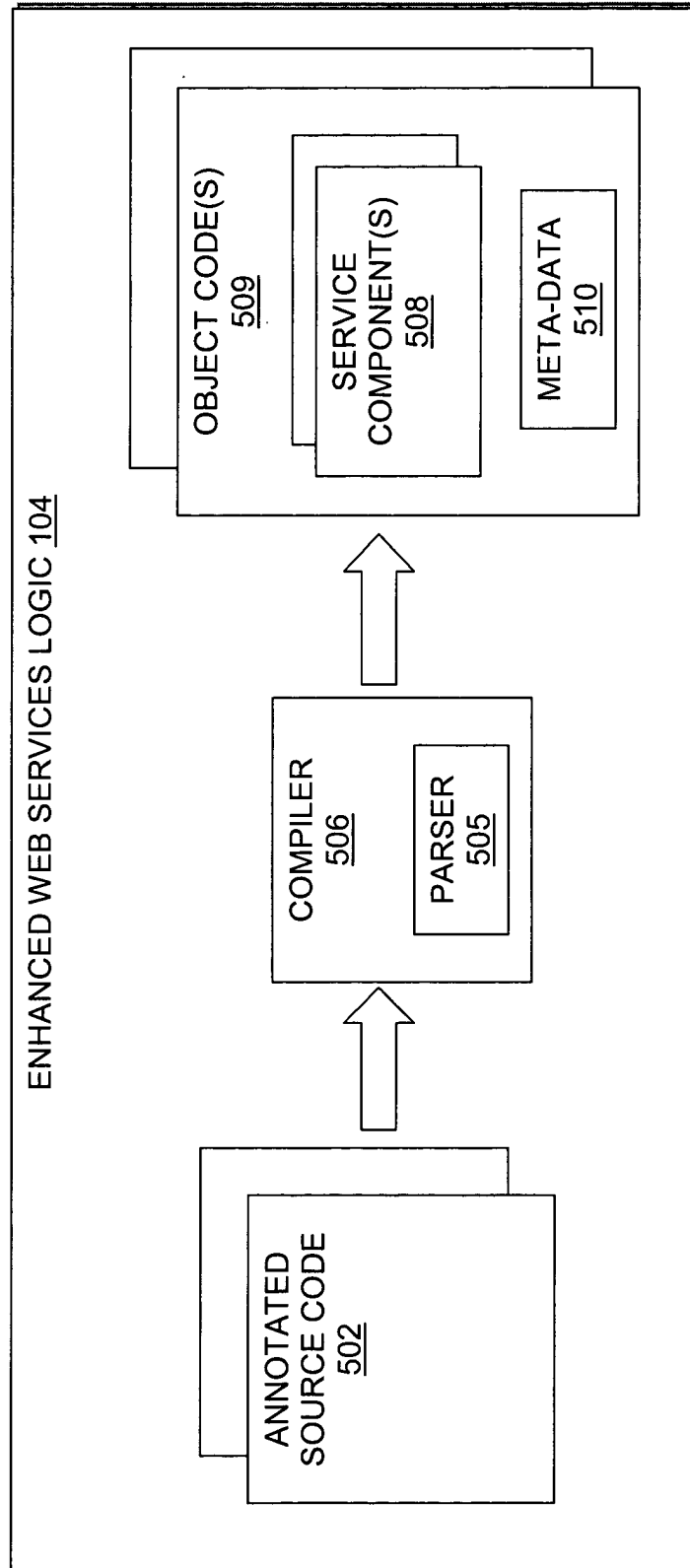


Figure 5

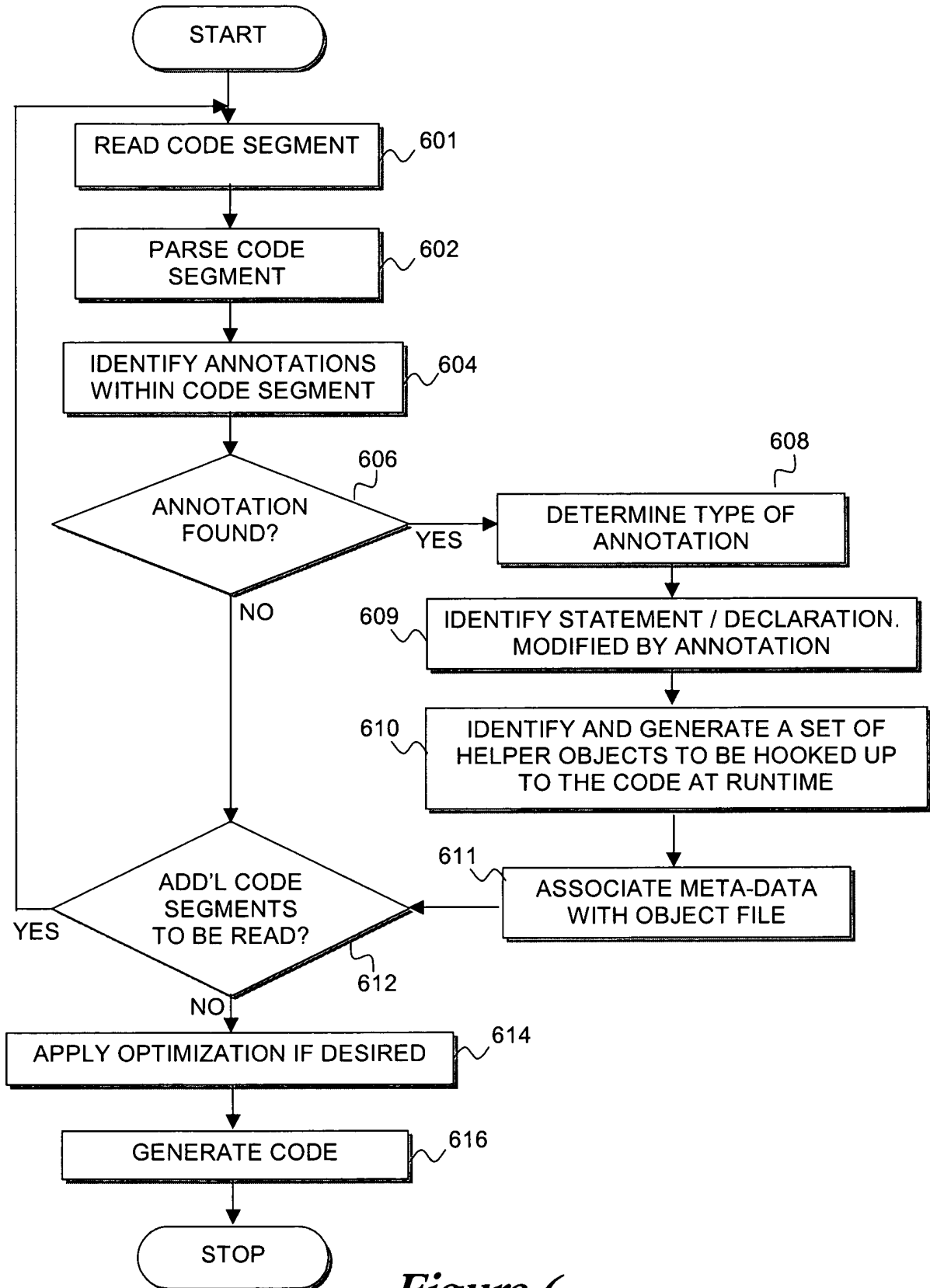


Figure 6

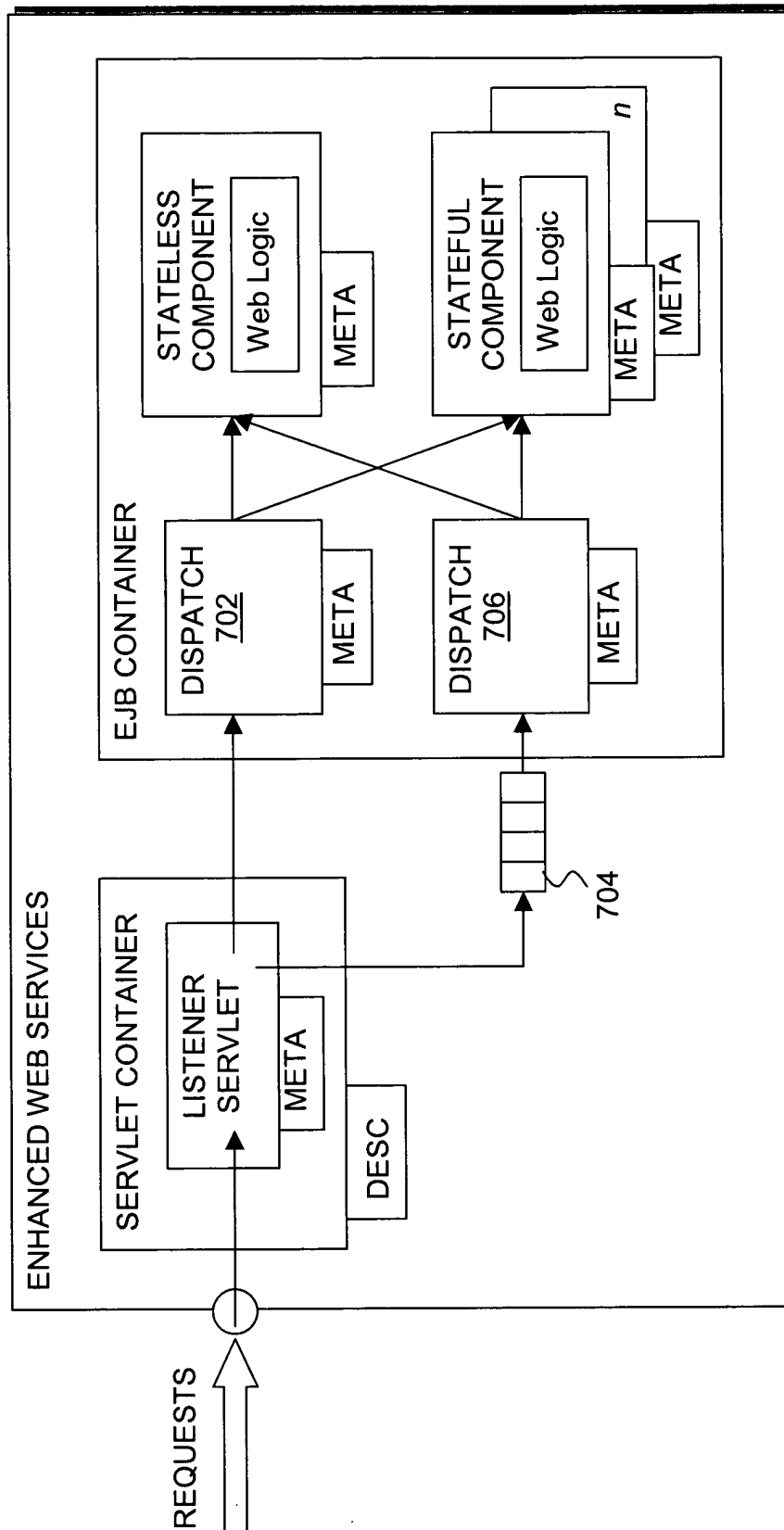


Figure 7

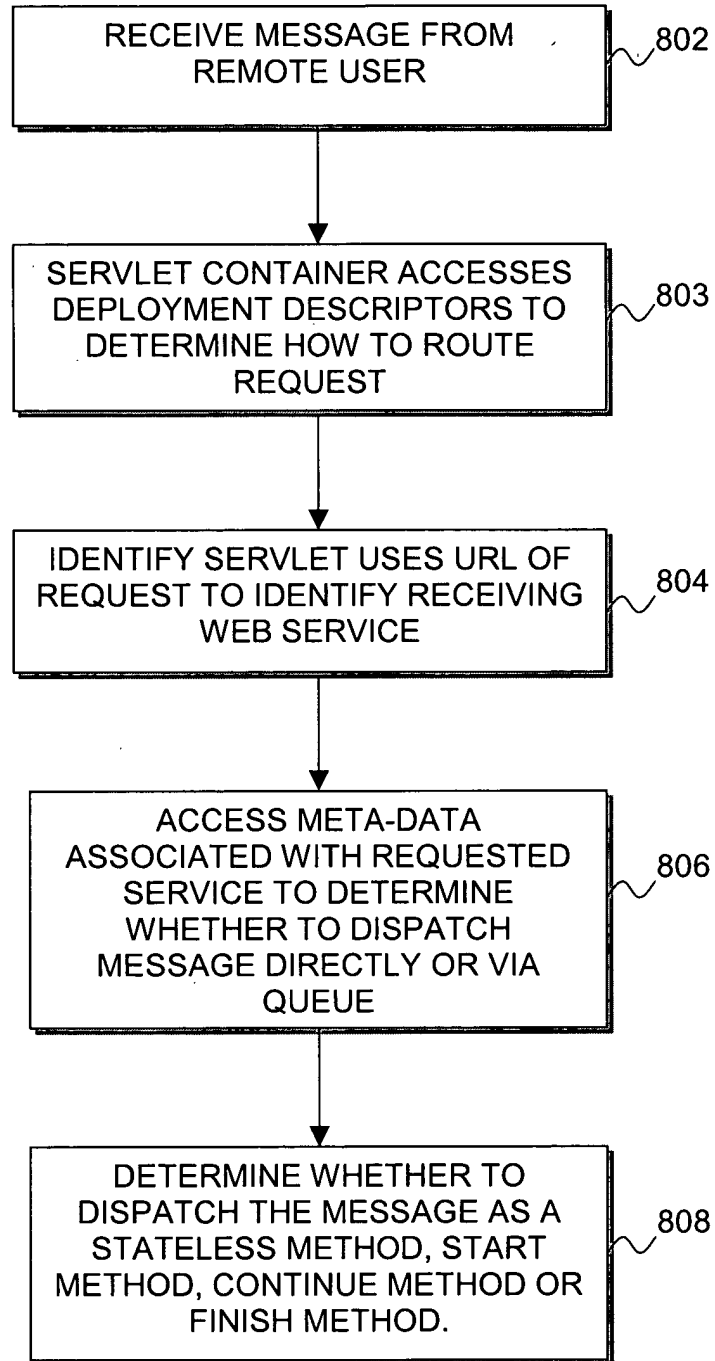


Figure 8

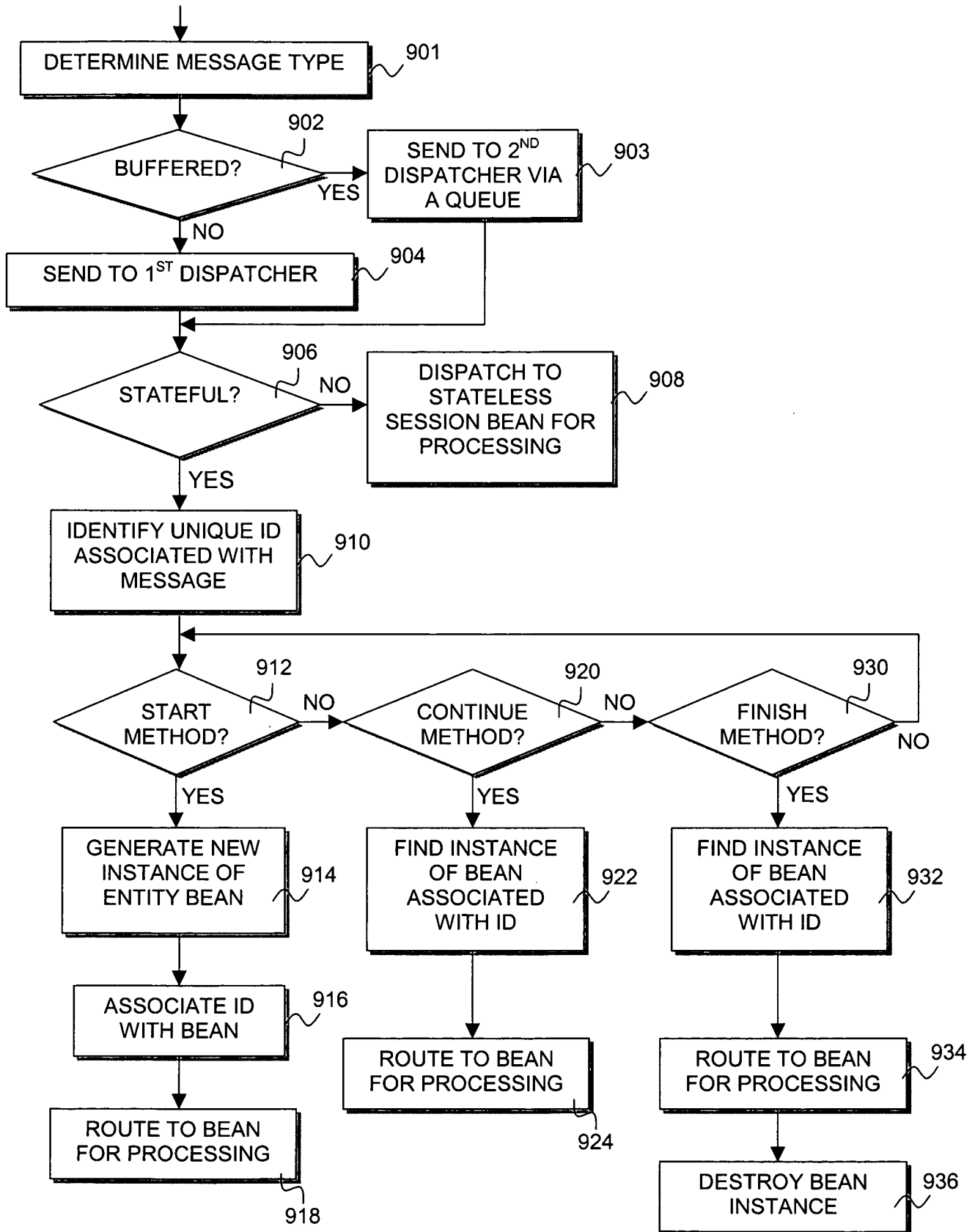


Figure 9

1000

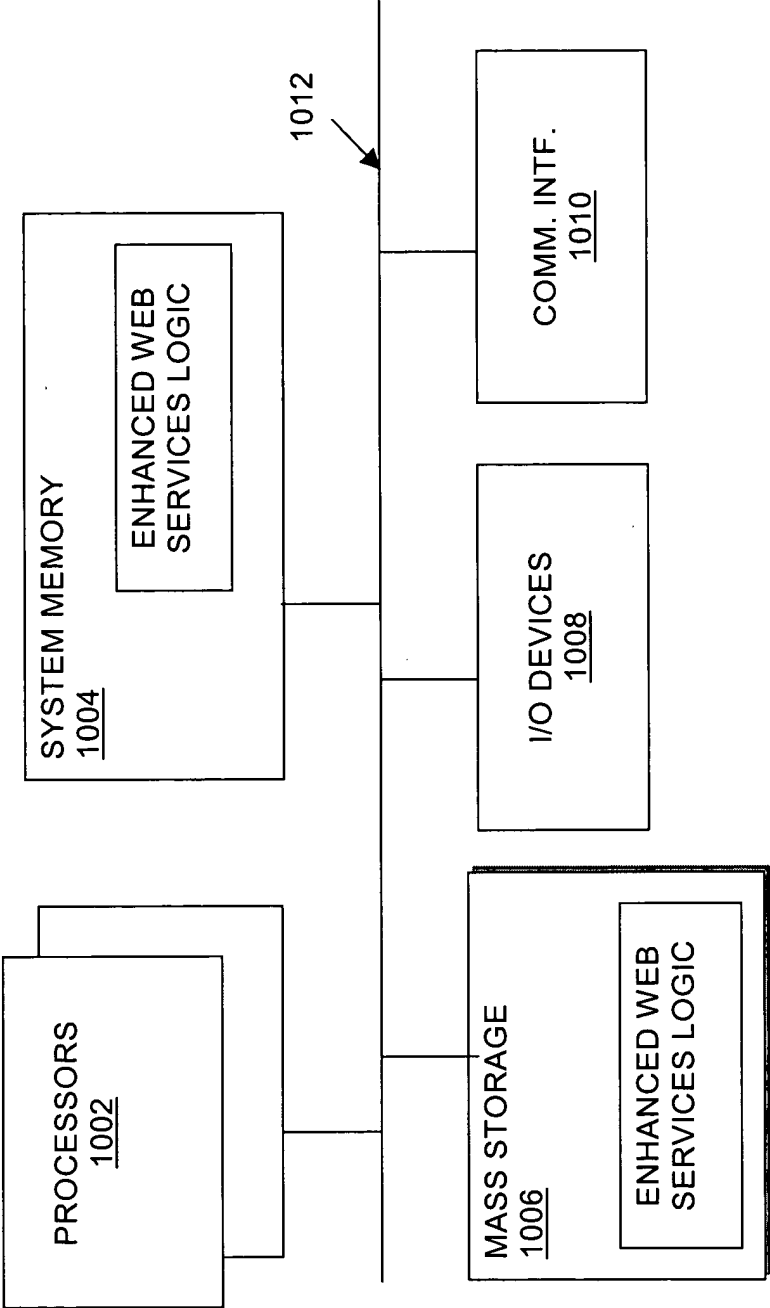


Figure 10

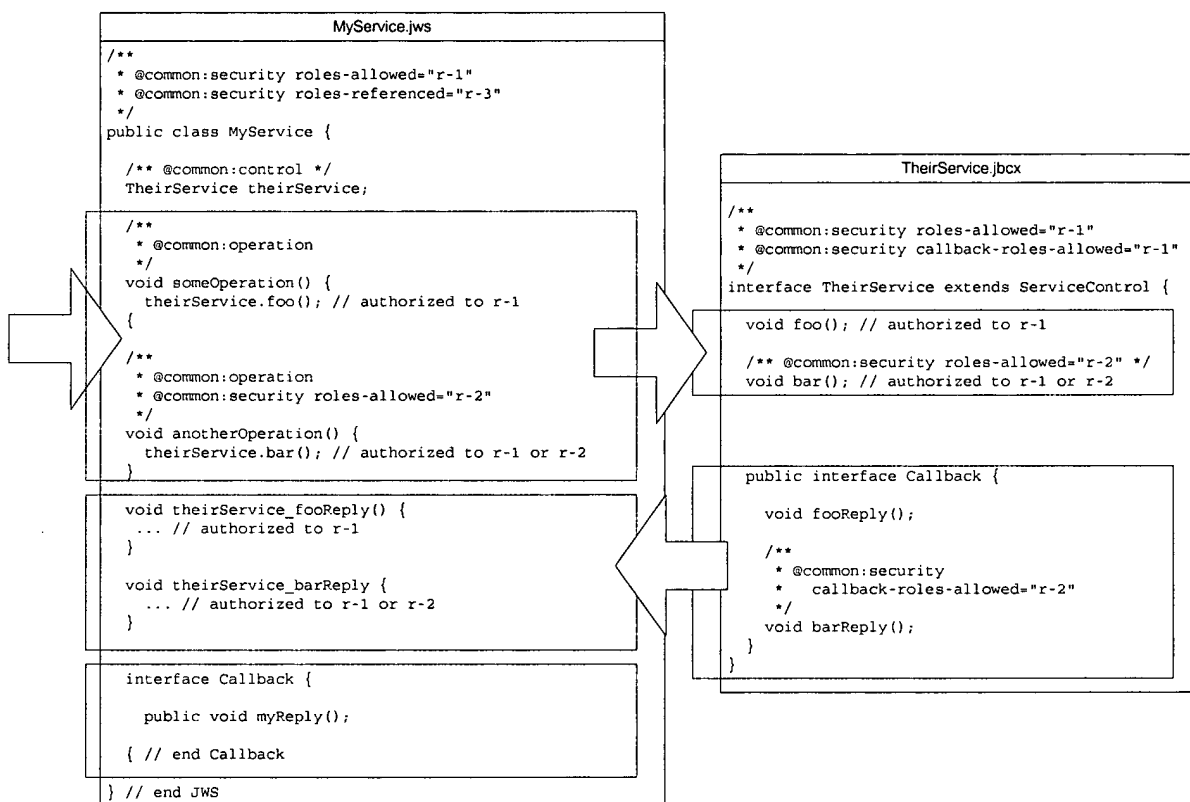


Figure 11

Using run-as="<start-user>" will cause the Subject that *started* the conversation to be bound to all continue/finish calls *from* the JW(X).

1. Invoke privileges on Control operations will be evaluated with respect to the roles granted to the starting Principal
2. From within called Controls:
 - (a) Any call to `getCallerPrincipal()` will return the starting Principal
 - (b) Any call to `hadRole()` will be evaluated with respect to the roles granted to the starting Principal

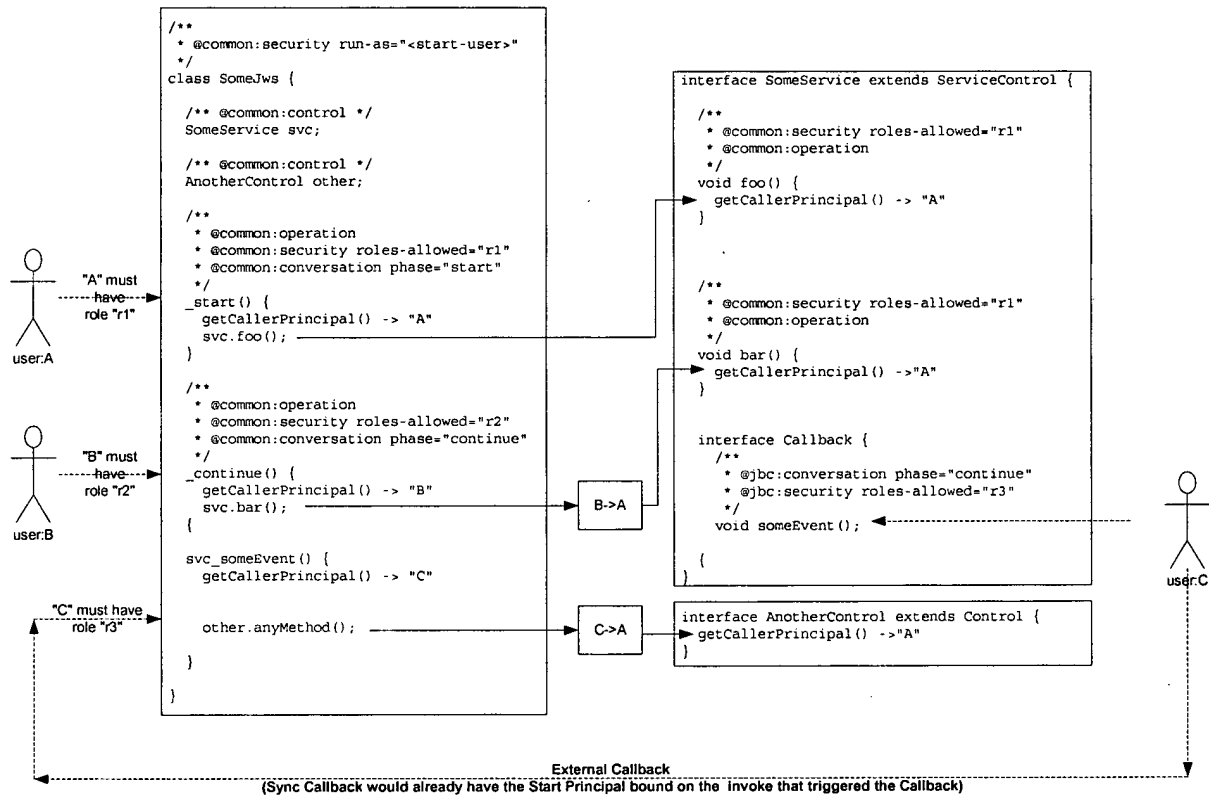


Figure 12

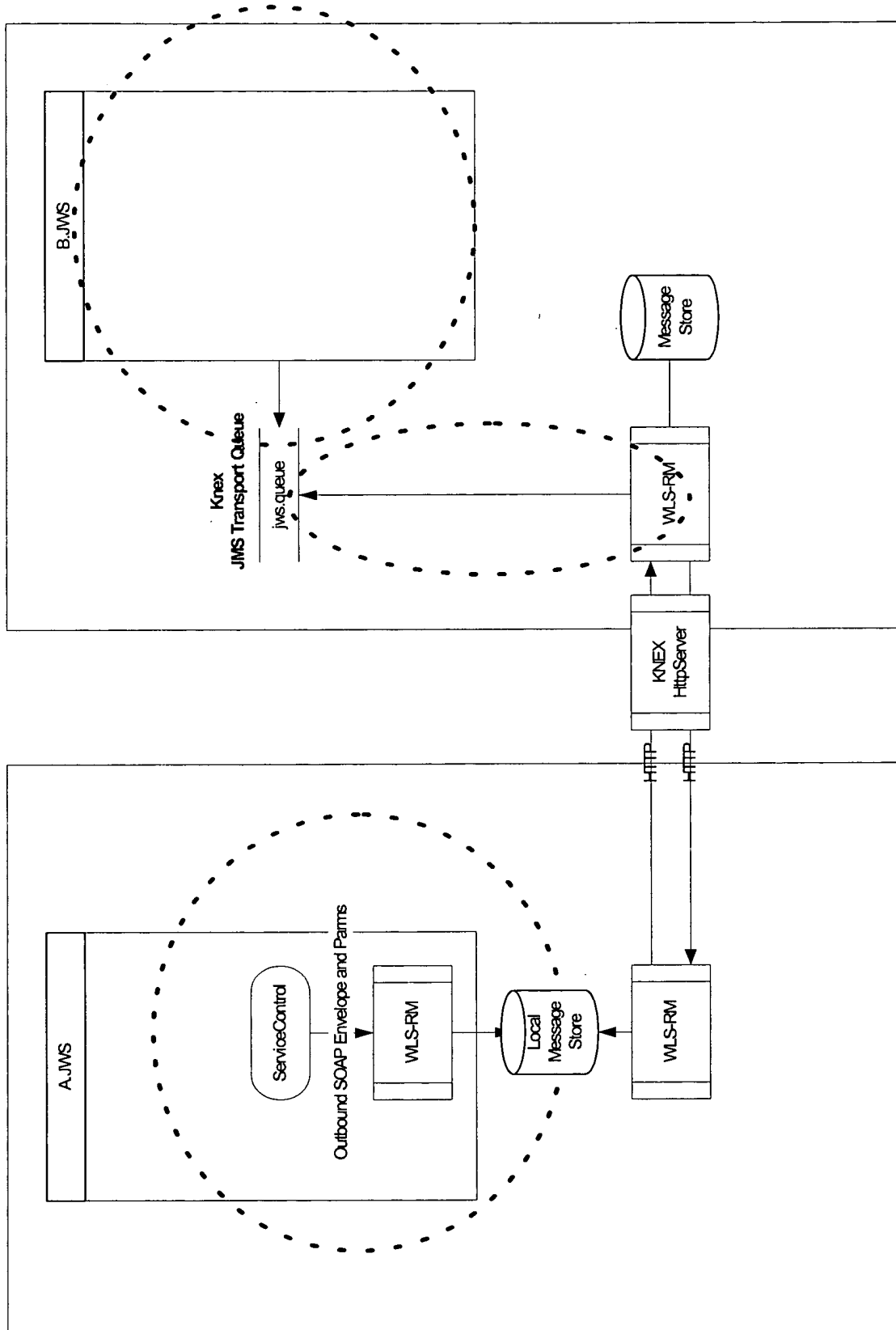


Figure 13



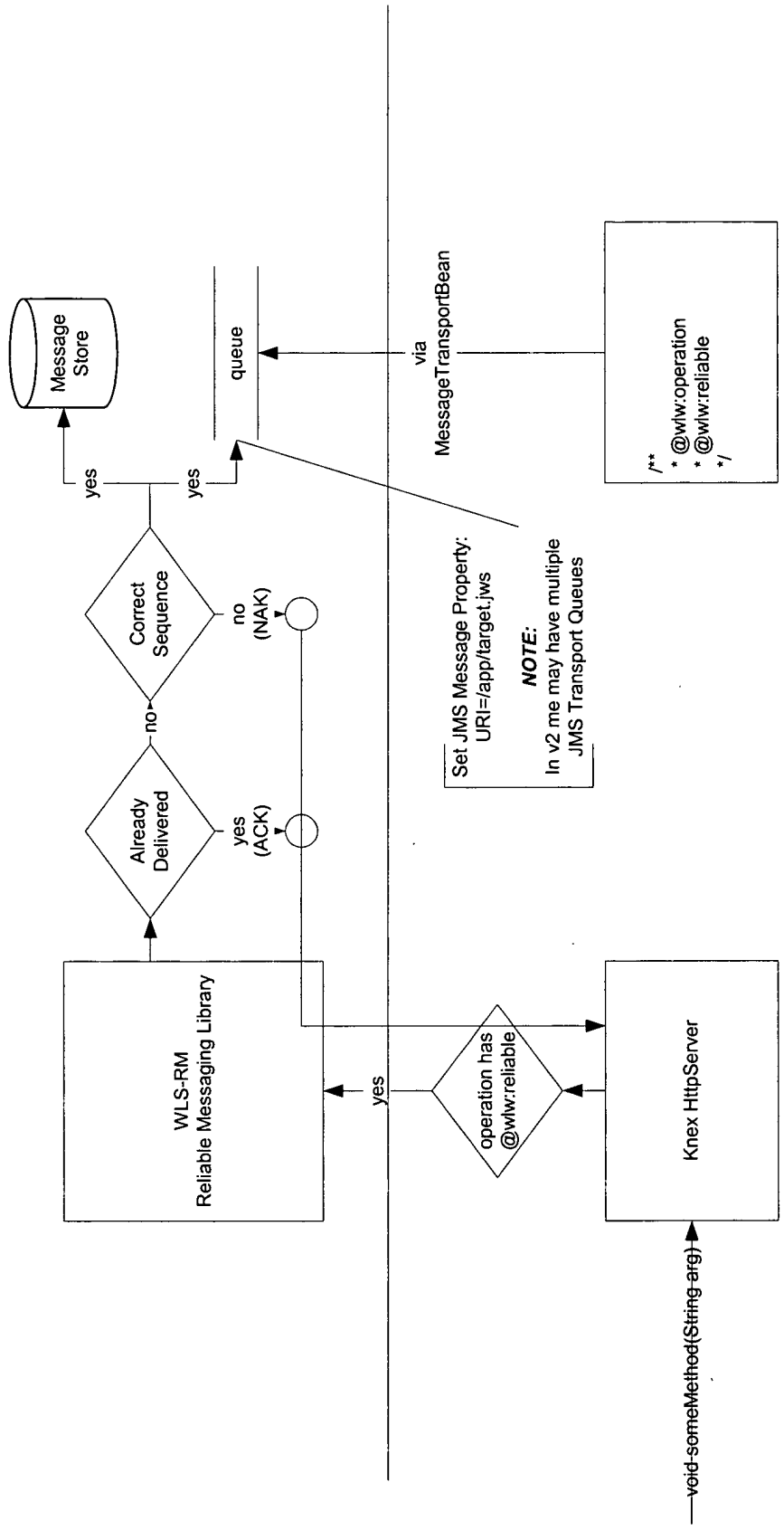


Figure 14

```
public interface ServiceInterceptor {  
  
    /**  
     * Provides initialization data to the interceptor implementation. This method  
     * will be called before any call to "handle" methods.  
     * @param config configuration parameters defined through config-data annotations  
     * @param wsdl the definition for this service  
     */  
    public void init(Map config, XMLInputStream wsdl);  
  
    /**  
     * The interface called from the knex runtime to allow top-level  
     * containers to interrogate and/or manipulate the incoming XML  
     * request. The request represents the entire payload of the  
     * request. This interface is invoked before any parameter-xml  
     * maps are run.  
     * @param xmlRequest The request that is targeted to an @operation  
     * method.  
     * @return an XMLInputStream that will be delivered to an @operation  
     * method on the current service.  
     */  
    public XMLInputStream handleRequest(XMLInputStream xmlRequest);  
  
    /**  
     * The interface called from the knex runtime to allow top-level  
     * containers to interrogate and/or manipulate the outgoing XML  
     * response. The response represents the entire payload of the  
     * response. This interface is invoked after any return-xml  
     * maps have run.  
     * @param xmlResponse The response that was generated by an @operation  
     * method.  
     * @return an InputStream that will be returned to the caller of the  
     * service.  
     */  
    public XMLInputStream handleResponse(XMLInputStream xmlResponse);  
  
    /**  
     * The interface called from the knex runtime to allow top-level  
     * containers to interrogate and/or manipulate the outgoing XML  
     * response. The response represents the entire payload of the  
     * response.  
     * @param xmlResponse The response that was generated by an @operation  
     * method.  
     * @return an InputStream that will be returned to the caller of the  
     * service.  
     */  
    public XMLInputStream handleFault(XMLInputStream xmlFault);  
  
    /**  
     * Called when the service instance is being removed. After destroy is called,  
     * none of the "handle" methods will be calls.  
     */  
    public void destroy();  
}
```

Figure 15